基于帕累托优化的网络安全设备部署设计与研究 *

冯　毅，潘　上，李　瑞
(信息工程大学，郑州 450000)

摘　要：通过部署网络安全设备可以有效地提高网络的安全性，但由于网络设备种类繁多、功能复杂，如何在整个网络中最优地部署网络安全设备，从而达到安全和开销的平衡，仍是研究人员关注的焦点。将网络安全设备最优部署问题转换为帕累托优化的问题，提出分布式约束优化的七元组对网络安全设备的部署进行量化赋值，构建基于分支界限算法的部署方案搜索算法，在解空间内对量化的数值进行计算并求出最优解。由于基于分支界限算法的方案搜索算法需要耗费大量时间，在大型网络中运行效率较低，使用基于弧一致优化的数据预处理技术对量化数值进行预处理，实现搜索算法的优化。最后通过仿真实验测试，证明所提出方法的正确性和有效性。

关键词：网络安全设备部署；帕累托优化；分支界限搜索算法；弧一致预处理技术
中图分类号：TP393.08　　doi: 10.19734/j.issn.1001-3695.2018.10.0887

Design and research of cyber security device deployment based on Pareto optimality

Feng Yi, Pan Shang, Li Rei
(*Information Engineering University*, *Zhengzhou* 450000, *China*)

Abstract: By deploying cyber security devices, cyber security can be effectively improved. But due to the varieties and complex function of cyber equipment, how to optimize the deployment of cyber security devices is still a core problem for researchers.By turning the deployment of cyber security equipment into a pareto optimization problem,this paper put forward a tuple, a distributed constraint optimization, to quantify the deployment of cyber security device. Then, the algorithm, based on branch and bound algorithm, is used to calculate the quantized values to find out the optimal solution within solution space. And it takes a lot of time to run the algorithm, based on branch and bound algorithm, which is infeasible in large networks. This paper proposes to use the data preprocessing technology to optimize the search, which is based on arc consistency technology. This paper computed the quantized value in advance and make the search become an easy problem. Finally, it used the simulation experiments to prove the correctness and effectiveness of the proposed approaches.

Key words: cyber security device deployment; Pareto optimization; branch and boundary algorithm; arc consistency technology

0　引言

通过部署网络安全设备提高网络安全性已成为广泛共识，但由于网络设备种类繁多、系统复杂、功能多样，如何在整个网络内最优地部署网络安全设备，在充分保证网络安全的同时，减少对网络性能的影响，降低经费、能源、管理等方面开销，仍是网络安全研究人员关注的焦点。文献[1]提出了基于内罚函数对约束优化问题进行求解。内罚函数中增广目标函数的 Hessian 矩阵与惩罚因子的大小关系密切，但如何设置初始的惩罚因子，仍需要耗费大量运算。文献[2]提出基于贪心算法构建最优部署方案，但如果时间有限，贪心算法只能形成局部最优。帕累托优化（Pareto improvement）是指在不牺牲任何一种变量的前提下，通过帕累托改进，使其他变量达到更优。网络安全设备优化部署问题是在各种部署方案中选择最优部署方案的问题，因此可以将其转换为帕累托优化问题。

为解决帕累托优化问题，研究人员设计了很多搜索算法。文献[3]利用权重系数法求逆变器多目标最优控制序列。该问题中权重系数较多依赖初始赋值，初始赋值的设定依赖人为经验，降低了该算法的效率。并且该文献没有对损耗定量计算。文献[4]运用免疫算法（immune algorithm, IA）用于优化控制三相逆变器，虽然解决了 GA 在收敛速度和局部搜索能力等方面的不足，但没有考虑多维目标问题。文献[5]提出了多目标粒子群优化算法，个体适应度评估决定了多目标粒子群优化算法中选择全局最优解和维护档案的这两个关键策略，要么是基于密度的评估来考察个体的多样性潜力，要么是基于 帕累托占优关系的评估来考察个体的收敛性潜力，因而导致在维护外部档案时无法统筹兼顾帕累托最优前端的多样性和收敛性，而在选择全局最优解时必须考虑的。

在本文中，选择使用分支界限算法的深度优先策略在解空间中搜索可能的部署方案。分支界限算法由一个合理的界限函数确定目标函数的界，并通过对孩子节点估值确定是否进行剪枝，可以有效提高搜索效率。但是，在含有众多节点的大型网络中仅仅使用分支界限算法执行优化依然需要大量时间，为此采用弧一致性算法的预处理技术对搜索算法进行扩展，将分支界限中目标函数的约束优化问题转换为一个可

收稿日期：2018-10-19；**修回日期：**2019-02-01　**基金项目：**国家自然科学基金资助项目；信息工程大学科研基金资助项目
作者简介：冯毅（1981-），男，湖北武人，副教授，博士研究生，主要研究方向为网络安全、人工智能与安全度量、网络安全防御（longer15@sina.com）；潘上（1996-），男，河南唐河人，助理工程师，主要研究方向为帕累托优化；李瑞（1996-），男，助理工程师，主要研究方向为机器学习与网络安全防御.

以被有效解决的问题，使得分支界限算法可以快速剪枝，大大提高了搜索效率。

1　模型与量化赋值

1.1　模型

在网络安全设备最优部署方案的求解过程中，首先要解决的是对网络安全设备最优部署问题的形式化描述，运用具有严密逻辑结构的形式化描述语言对网络安全设备最优部署问题进行抽象。本文使用多目标条件下的分布式约束优化方法（multi-objective distributed constraint optimization, MO-DCOP）来形式化地描述最优部署问题。

在此方法中，网络安全设备最优部署问题被描述为一个六元组问题，即 $<S,X,D,C,O,U,V>$，其中 $S=\{0,1,2,...,n\}$ 为网络中的节点集合，每个节点表示一个可以部署网络安全设备的位置（如软件系统、终端主机、路由器、网关等），$X=\{X_0, X_1, ..., X_n\}$ 为变量集合，$D=\{D_0,D_1,...,D_n\}$ 为离散域集合，每一个节点 i 都有其对应的变量 X_i，它的值取决于离散域 D_i=*{设备部署，设备不部署}*。$C=\{C_1, C_2\}$ 为约束集合，　$O=\{O_1, O_2\}$ 为目标函数集，C_1, C_2 分别表示负载指数和风险指数的约束集，O_1, O_2 分别表示负载开销指数和网络风险指数的目标函数集。$U=\{U_1,U_2\}$ 为权重系数集，U 表示对于目标函数 O 所占权重值，其值取决于网络管理员对约束目标的关注程度。$V=\{V_1, V_2...,V_m\}$ 表示各个安全部署方案中负载指数和风险指数的集合。

由于网络中的节点并不是单独存在，在一个节点上部署网络设备可能会对另一个相连节点也产生影响。因此对于网络中的两个相连节点 i, j，（i, j）表示两个节点所对应变量 X_i 和 X_j 之间存在着约束关系，对于目标 l $(1 \leqslant l \leqslant 2)$)和存在约束关系的变量 X_i，X_j 而言，每种抉择 $\{(X_i,D_i),(X_j,D_j)\}$ 所带来的负载指数和风险指数的变化通过函数 $f_{i,j}^{l}(d_i,d_j):D_i \times D_j$ 来描述，考虑某一目标 l，对于所有变量 X 及其所有抉择方式 A，综合考虑后表示为

$$R^{l}(A)=U_l * \sum_{(i,j)\in C^{l},\{(X_i,D_i),(X_j,D_j)\}\sqsubset A} f_{i,j}^{l}(d_i,d_j)$$

基于此，网络安全设备最优部署问题可以被表示为：$\mathrm{R}(A)=\{R^1(A),R^2(A)\}$。对于这个问题，理想的网络安全设备部署方案就是使得网络中综合负载指数与风险指数均达到最小值。通常，这样的解决方案需要通过大量的计算和设计者丰富的经验来获得，实现起来难度较大。因此，通过引用帕累托最优化的思想，本文通过智能计算在解空间中找到一个最佳的解决方案。

定义 1　帕累托最优决策。对于网络安全最优部署问题和所有节点 A 的部署决策，称 A 是帕累托最优决策，当且仅当不存在其他决策 A'，使得 $R(A') \prec R(A)$。也称 $R(A)$ 支配 $R(A')$。

1.2　量化赋值

在形式化描述最优部署问题之后，接下来就需要对模型中的约束集和权重系数集进行量化赋值。由于网络态势描述了各种网络设备运行状况、网络行为以及用户行为等因素所构成的整个网络的当前状态和变化趋势，本文中使用网络态势中的相关要素对约束集中的属性进行定义。借鉴网络态势中基于指标体系的评估方法对网络约束集中属性进行描述，使用网络态势感知技术，对部署网络安全设备之后网络态势发生变化的要素进行获取、理解、评估，并对约束集中的属性进行量化。在参考现有网络安全态势评估方法的基础上，本文提出了使用节点安全评估指标，主要从负载开销和网络风险指数两方面展开。

定义 2　负载开销。负载开销主要从网络节点的软/硬件能力和实际负载角度出发，考虑其在部署安全设备后是否能够健康运行，能否正常向用户提供服务，以及部署后对网络的影响。负载开销主要通过以下指标进行量化计算：

a)峰值流量。峰值流量是两个相连的网络节点在一定的时段内传输的最大数据率，用来衡量该网络设备安全状况最严重的情形。

b)带宽利用率。两个相连的网络设备进行通信时对于网络带宽的利用率。较高或较低的带宽利用率会降低网络的性能并限制网络中的正常活动。

c)CPU 利用率。取两台相连的网络设备中 CPU 利用率高的那个值，CPU 利用率越高说明对资源的占用越多，完成其他任务的能力越弱。

d)内存利用率。与 CPU 利用率类似，内存利用率用来衡量两个相连的网络设备的实时性能。

e)网络安全设备费用。这一指标主要衡量部署网络安全设备后在经费、能源、管理等方面的开销。

针对上述负载指标的量化主要按照如下的步骤进行：

a)在某个时间段内用各个指标的过载率来衡量严重程度，过载率如下定义：

$$OL_{\mathrm{k}\,i,\,j} = \begin{cases} \dfrac{l_{\mathrm{k}\,i,j}}{L_{\mathrm{k}\,i,j}} & if\ l_{\mathrm{k}\,i,j} > L_{\mathrm{k}\,i,j} \\ 1? & if\ l_{\mathrm{k}\,i,j} \le L_{\mathrm{k}\,i,j} \end{cases}$$

其中: $k \in \{1, 2, 3, 4\}$ 分别表示流量峰值、带宽利用率、CPU 利用率和内存利用率四个属性，i，j 是两个相连节点，$L_{k(i,j)}$ 代表属性 k 在节点（i, j）部署网络安全设备之前的统计值，它的取值符合各个时间段的统计规律，$l_{k(i,j)}$ 代表了属性 k 在节点（i, j）的真实值。$OL_{k(i,j)}$ 是属性 k 在节点（i, j）的过载率。对于网络安全设备费用指标，其计算方法是将设备一段时间内的费用（包括经费、能源、管理）除以同类设备的统计平均值，即 $L_{5(i,j)}$ *表示*节点（i, j）所部署网络安全同类设备的费用平均值。

b)对过载率进行量化，本文将其按照规则将过载率分为 5 个等级。以 $Q_{k(i,j)}$ 表示 $OL_{k(i,j)}$ 的量化结果，$Q_{k(i,j)}$ 的计算过程如表 1 所示。

表 1　过载率量化表

Table 1　Quantification of overload rate

属性 k	$OL_{k(i,j)}$	$Q_{k(i,j)}$
当 $k \in \{1, 2, 3, 4\}$ 时	$OL_{k(i,j)} = 1$	0
	$1 < OL_{k(i,j)} \le 1.25$	1
	$1.25 < OL_{k(i,j)} \le 2$	2
	$2 < OL_{k(i,j)} \leqslant 3$	3
	$OL_{k(i,j)} > 3$	4
当 $k = 5$ 时	$OL_{k(i,j)} = 1$	1
	$1 < OL_{k(i,j)} \le 1.25$	2
	$1.25 < OL_{k(i,j)} \le 2$	3
	$2 < OL_{k(i,j)} \leqslant 3$	4
	$OL_{k(i,j)} > 3$	5

对于节点（i, j），该节点的负载指数可以表示为

$$\mathrm{Q}_{(i,?j)} = \sum_{k=1}^{5} Q_{k(i,?j)} \times u_{(i,?j)}$$

其中:u_k 表示属性 k 的权重值，该值由网络安全设备部署方案设计人员根据网络应用环境及其性能需求提前设定。负载开销的量化过程还可以根据实际网络环境灵活设置其他负载指标和量化规则。

定义 3　网络风险指数。网络风险指数被用来评估网络

攻击对网络节点所带来的影响，指数越大说明该网络节点受到网络攻击的风险越高、后果越严重。网络风险指数主要通过以下指标量化计算：

a)节点资产。在国际标准 ISO/IEC 13335 中任何对组织的有价值的实体被定义为资产。资产价值越高的网络节点，受到网络攻击的风险也就越大。

对资产的量化，采用国际标准 ISO/IEC 13335 中的定义来衡量网络节点的重要程度。其定义的等级如下：1 级为“可忽略的”；2 级为“低”；3 级为“中”；4 级为“高”；5 级为“严重”。

b)安全级别。在安全态势级别[1]分类的基础上，结合 CC 标准，本文将网络设备级别分为七级并进行等级赋值。

网络设备的安全级别对应 CC 标准的安全可信度级别，即网络设备所处的安全级别需要使用 CC 标准对应级别的安全产品来满足其安全需求。安全级别的值越高说明网络设备安全性越差，风险越高，需要使用的安全产品越复杂、功能越全面。

网络风险指数的计算相对简单一些其公式如下：

$$RS_{(i,j)} = \sum_{t=1}^{2} RS_{t(i,j)} \times p_t$$

其中：$RS_{(i,j)}$表示节点 i 和节点 j 的网络风险指数，$RS_{t\,(i,j)}$表示节点 (i,j) 风险指标 t 的量化值，p_t 表示指标 t 的权重。

然而在实际情况下，将所有网络设备部署到网络中，运用网络态势感知技术进行量化评估是不现实的。针对这个问题，研究人员提出了一些行之有效的解决办法。本文中使用了最简单的两个办法：a)通过构建相同或相似的网络结构，将需要量化的网络安全设备分别部署于网络中，运用态势感知技术对其相关要素进行量化评估，最终计算出约束值；b)对于无法构建真实网络环境的情况，运用仿真技术构建其网络环境，随后在仿真环境中输入安全设备的相关参数，模拟安全设备的部署，通过仿真技术和态势感知技术对仿真网络环境中的要素进行量化评估计算出约束值。这两个方法虽然无法度量出网络安全设备部署时的准确约束值，但通过实际部署和模拟仿真，在一定程度上能接近其真实的约束值并反映出设备部署网络的影响。因此所量化得到的结果在一定程度上能够满足进行设备部署优化时的计算需求。

1.3　网络安全设备安全保障能力量化

不同类别的安全设备，它们提供安全保障能力有所不同，因此需要对各类网络安全设备的安全特性进行量化评估，以便更好的完成设备部署。目前对于信息安全产品通常采用攻击性检测、软件工程质量评估、用户评价等方法进行分析，主要采用 CC 安全性评估准则标准进行度量。CC 标准将信息安全产品的安全可信度分为七个安全级别，将安全功能需求分为了 11 类。在参考网络与信息安全相关标准深入分析网络安全特性的基础上，本文初步定义七个网络安全需求，并将基与 CC 标准安全功能需求和网络安全设备之间建立起对应关系，如图 1 所示。

按照图 1 中的对应关系，本文将网络安全设备的安全保障能力对应到 CC 标准中的安全功能上，并结合 CC 标准的安全功能需求进行度量。按照此方法对每个网络安全设备的安全保障能力水平重新进行计算，得到网络安全设备安全保障能力量化数据。量化过程如下：首先确定该网络安全设备所能满足的网络安全需求，随后利用图中的对应关系，找出该网络安全设备所能满足的 CC 标准安全功能需求，最后根据 CC 标准安全功能需求确定其安全级别，该安全级别即为其量化值。

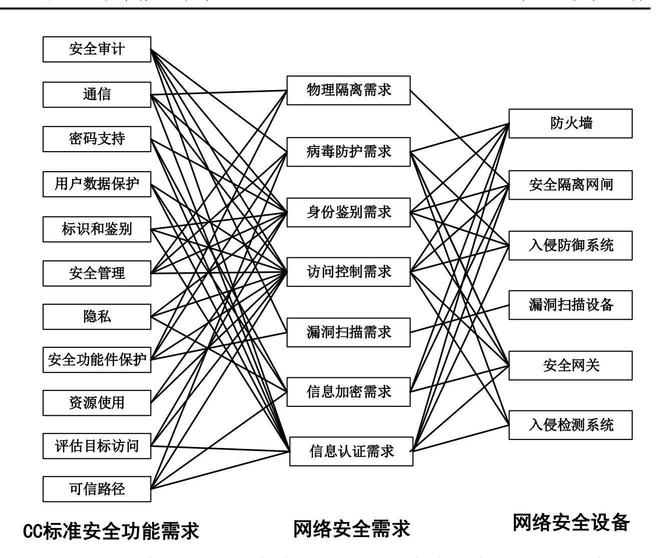


图 1　CC 标准、网络安全需求、网络安全设备之间的关系

Fig. 1　Relation between CC, network security needs and network security equipment

2　基于分支界限算法的部署方案搜索

2.1　分支界限树

基于分支界限算法的解空间搜索技术，本文将网络安全设备方案部署问题转换为在分支界限树上最优路径的搜索问题。树的每一层节点表示网络中某一节点上的安全设备部署方案，其左孩子节点表示部署方案中在该节点上部署网络安全设备，右孩子节点表示不部署。因此，任意有连接关系的两个节点之间有四种状态（部署，部署），（不部署，不部署），（部署，不部署），（不部署，部署），对应四个约束值。自上而下的某一条路径代表一种部署方案，并在叶子节点处计算部署该方案的约束值。整个分支界限树包含了全部的部署方案。

本文以三个节点的网络为例进行说明（网络中的节点分别表示为 0、1、2，并相互连接），图 2 表示其分支界限树。

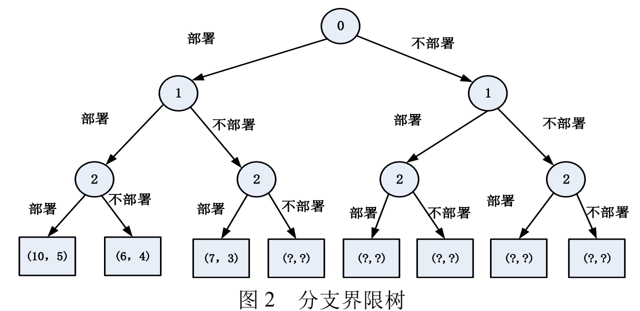


图 2　分支界限树

Fig. 2　Branch and bound tree.

例如，最左侧一条路径中表示 0，1，2 三个节点上均部署网络安全设备，叶子节点上的值表示当采用所有节点都部署网络安全设备的部署方案时，对应网络风险指数和负载开销指数分别为 10 和 5。

2.2　部署方案搜索算法

对部署方案的搜索过程，实际上也就是对方案约束值进行优化比较的过程。叶子节点上的变量值表示选择对应部署方案进行网络安全设备部署时，网络风险指数和开销负载指数的值 V。初始目标函数值 $O=\{(\infty,\infty)\}$，当 $V \prec O$ 时，表示该路径优于当前目标函数值，对该目标函数值 O 进行更新，清空目标函数集合 O，并将 C 值输入目标函数集合 O；当有 $O \prec V$ 时，继续搜索下一条路径；当 $O \prec V$ 与 $V \prec O$ 均不成立时，不清空目标函数集合 O，将 C 值加入目标函数集合 O。

最终目标函数 O 中的值即为最优部署方案的约束值，其对应部署方案即为最优部署方案。

普通的搜索算法是对所有叶子节点进行计算，然后一一比较，得出最优部署方案。分支界限算法由一个合理的界限函数确定目标函数的界，并通过对孩子节点估值确定是否进行剪枝，可以有效提高搜索效率。基于分支界限法的部署方案搜索算法如算法 1 所示。

算法 1　网络安全设备最优部署方案搜索算法

Input：Tuple $\langle S,X,D,C,O,U,V\rangle$, quantitative assignment for each pair nodes.

Output：Object function O.

Initialize　$O \leftarrow \{(\infty,\infty)\}$

Stage 1:

build branch-bound tree

Repeat

Stage 2:

run depth-first traversal in branch-bound tree;

Stage 3:

For $k=1$　$k\leqslant n+1$　$k=k+1$ do

compute v_k for the node k in traversal path,

$$v_k=\left(\sum_{(i,j)\in S?} Q_{(i,?j)},?\sum_{(i,j)\in S} RS_{(i,j)}\right)$$

$0\leqslant i<j\leqslant k$;

if $O\prec v_k$ then

　　run Stage 2;

else if $k=n+1$ then

　　if $v_k\prec O$ then

　　　　$O\leftarrow V=v_k$;

　　else then

　　　　$O\leftarrow O\cup v_k$;

　　end if

end if

end if

Until all the leaf nodes are traversed

算法首先初始化目标函数集合 $O=\{\infty,\infty\}$。随后使用分支界限进行深度优先遍历搜索，对搜索路径中的每个节点 k 计算从根节点到该节点所产生约束 v_k。将 v_k 与目标函数集合 O 进行比较，若 $O\prec v_k$，则对其进行剪枝处理，否则，继续深度优先进行遍历。$k=n+1$ 时，即搜索到叶子节点，此时网络安全设备部署方案的风网络险指数和开销负载指数的值 $V=v_k$，约束集合 C 和目标函数集合中的元素进行比较。如果 $V\prec O$，则 V 覆盖 O，否则，只将 V 压入目标函数集合 O，继续遍历。当遍历所有叶子节点后，目标函数集合中的元素即为搜索出的最优部署方案。

3　基于弧一致预处理的搜索算法优化

在含有众多节点的大型网络中仅使用分支界限算法执行优化依然需要大量时间，为此本文采用弧一致性算法的预处理技术对搜索算法进行扩展，将分支界限中目标函数的约束优化问题转换为一个可以被有效解决的问题，使得分支界限算法可以快速剪枝，大大提高了搜索效率。

3.1　基于弧一致优化的数据预处理

为解决分支界限算法的快速剪枝问题，本文利用基于弧一致优化的数据预处理技术对分支界限算法进行扩展。弧一致预处理技术是一种数据处理过程，从底层节点开始，自下而上传递每个目标的最小值。在网络安全设备部署问题中基于弧一致优化的数据预处理技术的结构如图 3 所示。每一层分别表示分支界限树上的一个节点，每一层包含两个取值点，分别表示在该节点部署或不部署网络安全设备的情况。不同层间的取值点连接形成约束，表示某种部署决策，连接上的值为两个节点之间四种部署状态对应四个约束值。

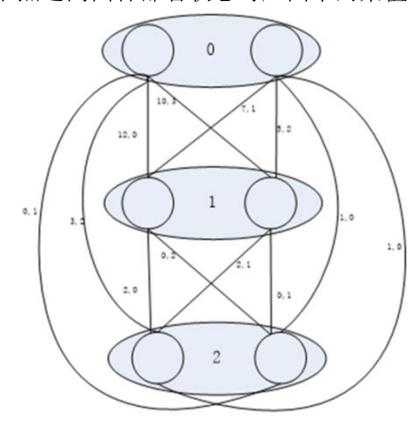


图 3　基于弧一致优化的数据预处理技术结构

Fig. 3　Data structure of preprocessing technology based on arc consistency technology.

基于弧一致优化的数据预处理技术是一种自下而上的数据处理过程，从叶子节点开始，每一层依次向上层传递理论中的最小值，如算法 2 所示。

算法 2　基于弧一致优化的数据预处理算法

Input：Tuple $\langle S,X,D,C,O,U,V\rangle$, quantitative assignment for each pair nodes, branch-bound tree.

Output:A set of variables $P=\{(p_0^1,p_0^2),(p_1^1,p_1^2),\ldots,(p_n^1,p_n^2)\}$

Initialize　$O\leftarrow\{(\infty,\infty)\}$,　$P=\{(0,0),(0,0),\ldots,(0,0)\}$;

for $k=n$　$1\leq k$　$k=k-1$

stage 1

$$p_{k-1}^1=\left(\min_{D_{k-1}=(deploy)} Q_{(k-1,k)},\min_{D_{k-1}=(deploy)} RS_{(k-1,k)}\right)+p_{k-1}^1\ ;$$

$$p_{k-1}^2=\left(\min_{D_{k-1}=(undeploy)} Q_{(k-1,k)},\min_{D_{k-1}=(undeploy)} RS_{(k-1,k)}\right)+p_{k-1}^2\ ;$$

$$p_{k-2}^1=\left(\min_{D_{k-1}=(deploy)} Q_{(k-2,k)},\min_{D_{k-1}=(deploy)} RS_{(k-2,k)}\right)+p_{k-2}^1\ ;$$

$$p_{k-2}^2=\left(\min_{D_{k-2}=(undeploy)} Q_{(k-2,k)},\min_{D_{k-2}=(undeploy)} RS_{(k-2,k)}\right)+p_{k-2}^2\ ;$$

…

$$p_0^1=\left(\min_{D_0=(deploy)} Q_{(0,k)},\min_{D_0=(deploy)} RS_{(0,k)}\right)+p_0^1\ ;$$

$$p_0^2=\left(\min_{D_0=(undeploy)} Q_{(0,k)},\min_{D_0=(undeploy)} RS_{(0,k)}\right)+p_0^2\ ;$$

$P=\{(p_0^1,p_0^2),(p_1^1,p_1^2),\ldots,(p_n^1,p_n^2)\}$;

stage 2

if $D_{k-1}=(deploy)$ and $2\leqslant k$ then

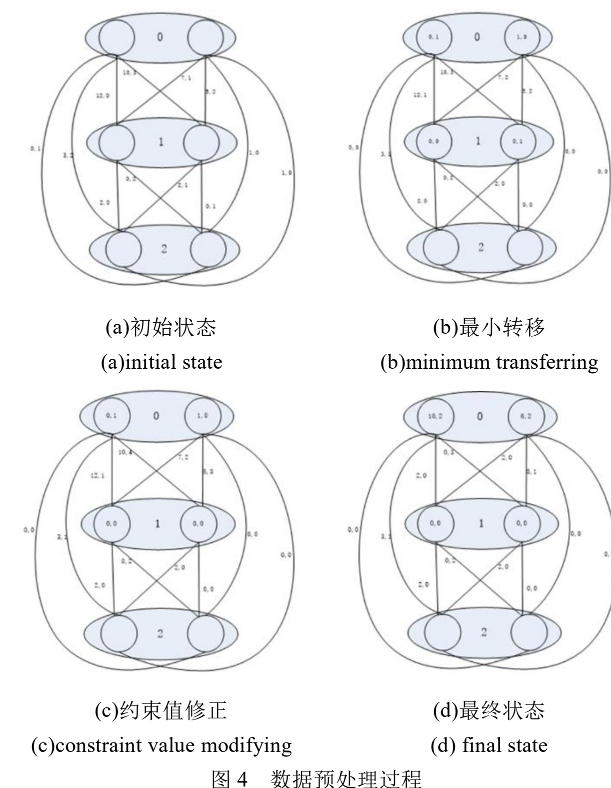
$(Q_{(k-2,?k-1)},?RS_{(k-2,?k-1)})=(Q_{(k-2,?k-1)},RS_{(k-2,?k-1)})+p_{k-1}^1$;

else $D_{k-1}=(undeploy)$ and $2\leqslant k$ then

$(Q_{(k-2,?k-1)},?RS_{(k-2,?k-1)})=(Q_{(k-2,?k-1)},RS_{(k-2,?k-1)})+p_{k-1}^2$;

end if

先将最下一层节点连线的最小约束值传递到对应节点的两个取值点中。然后将倒数第二层中两个取值点的值分别加入与倒数第三层两个取值点连接的约束值中。依次向上传递最小约束值，最终输出每一层中两个取值点的值，即为该节点部署方案中理论上的最小约束值。图 4 中的四张图分别展示，运用该算法对图 1 中的分支界限树的数据预处理过程。



(a)初始状态　　(b)最小转移

(a)initial state　　(b)minimum transferring

(c)约束值修正　　(d)最终状态

(c)constraint value modifying　　(d) final state

图 4　数据预处理过程

Fig. 4　The process of data preprocessing.

3.2　基于弧一致优化的分支界限算法

通过弧一致预处理技术可以准确获得每个节点部署方案约束值的下限，通过将已计算出的部署方案约束值与节点的约束值的下限进行比较，可以快速进行剪枝，从而有效检索部署方案。例如通过弧一致预处理算法求出某个节点 i 不部署时约束值下限为(10,2)，即节点 i 不部署网络安全设备的所有部署方案，其约束值不会小于(10,2)。如果此时分支界限算法计算过程中求得某个部署方案的约束值为(7,1)，(7,1) $\prec$ (10,2)，可以认为节点 i 不部署设备的解空间里没有最优解，从而直接剪去分支界限树中的所有右分支。基于弧一致优化的分支界限算法如算法 3 所示。

算法 3　基于弧一致优化的分支界限算法

Input: Tuple $\langle S,X,D,C,O,U,V\rangle$, quantitative assignment for each pair nodes.
Output: Object function O.
Initialize　$O \leftarrow \{(\infty,\infty)\}$ k=1 t=o;
stage 1
build branch-bound tree;
run Algorithm2,
output $P=\{(p_0^1,p_0^2),(p_1^1,p_1^2),\ldots,(p_n^1,p_n^2)\}$;
stage 2
run depth-first traversal in branch-bound tree, traverse the left branch of all the nodes, $D=\{D_0, D_1, \ldots, D_n\}=\{deploy, deploy, \ldots, deploy\}$, and compute v_n for the node n

$$O \leftarrow V = v_n = \left(\sum_{(i,j)\in S?} Q_{(i,?j)}, ? \sum_{(i,j)\in S} RS_{(i,j)} \right) \quad 0 \leqslant i<j \leqslant n;$$

repeat
stage 3
For t=o n-1 $\leqslant t$ t=t=1 do
if $O \prec f_t^2$ then
　cut all the branch that D_n=(undeploy);
end if
stage 4
run depth-first traversal in branch-bound tree;
for k=1 $k \leqslant n$+1 k=k+1 do
compute v_k for the node k in traversal path,

$$v_k = \left(\sum_{(i,j)\in S?} Q_{(i,?j)}, ? \sum_{(i,j)\in S} RS_{(i,j)} \right) \quad 0 \leqslant i<j \leqslant k;$$

stage 5
if $O \prec v_k$ then
　run Stage 4;
else if k=n+1 then
　if $v_k \prec O$ then
　　$O \leftarrow V = v_k$;
　else then
　　$O \leftarrow C \cup O$;
　end if
end if
end if
until all the leaf nodes are traversed

基于弧一致优化的分支界限算法可以提前计算出部署方案约束值的下限，缩小了分支界限算法中目标函数的比较范围，加快了分支界限树的剪枝速度，提高了解空间的搜索效率。

4　实验与结果分析

本文使用几种具有代表性的网络结构，测试文中提到的几种算法，计算其运行效率。通过对比，展示各个算法运行效率之间的区别，以及优化之后对于解空间搜索速度的提升。

4.1　实验环境与测试数据

所有实验均在自主搭建的 GPU 服务器上完成，GPU 为 GeForce GTX 1080 Ti，CPU 为 Intel Core i7-7700K、内存为 32 GB 的 DDR4。实验中，分别对星型、环型、树型、总线型、网状、混合型六种网络拓扑结构进行测试。每种结构中分别包含个 5、8、15 个节点，每一对相连节点的约束值 Q 和 RS 从 0 到 10 随机选择，从而构建约束集。由于是随机选择，所以与实际情况并不一定相符，但对于实验测试结果没有影响。在六种网络结构中，除了网状结构外，其他结构中并不是所有节点之间均存在连接，此时其对应的 Q 和 RS 定义取值为均 0。网状结构由于所有节点间均存在连接，因此其约束集如表 2 所示。

表 2　约束集取值

Table 2　The Value of Constraint Set

(D_i, D_j)	(0, 1)(0, 2)(0, 3)(0, 4)(1, 2)(1, 3)(1, 4)(2, 3)(2, 4)(3, 4)
(deploy, deploy)	(6, 3)(5, 4)(6, 5)(7, 5)(9, 4)(5, 4)(8, 2)(7, 4)(5, 5)(6, 3)
(deploy, undeploy)	(5, 7)(4, 6)(5, 5)(6, 5)(9, 5)(4, 7)(6, 3)(6, 5)(4, 6)(5, 3)
(undeploy, deploy)	(4, 6)(4, 8)(4, 6)(5, 7)(9, 5)(3, 8)(5, 6)(5, 4)(4, 7)(5, 4)
(undeploy, undeploy)	(3, 8)(2, 9)(4, 7)(2, 9) (8,6) (2, 6)(4, 9)(2, 7)(2, 8)(4, 7)

除了对比不同网络结构对算法运行效率的影响之外，本文还将分别测试在同一种网络结构中，不同节点数和不同连接数的情况下，各个算法的运行速率。通过对比，计算各算法在不同情况下的优化效率。为了节省时间，本文采用混合式的网络，其网络结构和节点间的连接随机分配。对于两个相连节点的约束值 Q 和 RS 同样进行随机选择，每一种情况

约束值取值五次，分别取出五组值约束集。针对这五组约束集，算法运行时间取五次的平均值。

4.2 不同网络拓扑结构测试与分析

为测试算法在不同拓扑结构中的运行效率，本文分别构建六种典型的网络拓扑结构，并随机设置节点间约束值。测试的算法主要有四个：普通搜索算法（GA）、分支界限搜索算法（BBA）、弧一致预处理算法（ACA）和基于弧一致优化的分支界限算法（ACA-BBA）。在本文实验中，这六种典型网拓扑结构分别包含有 5、8、15 个节点，由于拓扑结构不同每种网络中的连接数也不尽相同，不同结构中包含连接数如表 3 所示。

表 3 不同拓扑结构包含的连接数

Table 3 The Number of Connections Contained in Different Topologies

number of nodes	5 nodes	8 nodes	15 nodes
star topology	4	7	14
tree topology	4	7	14
ring topology	5	8	15
bus topology	4	7	14
hybrid topology	8	20	57
mesh topology	10	28	105

不同网络结构的测试分析如图 所示，图 5 中展示了对于不同算法在各种网络拓扑结构中运算速度的对比。图 5(a)~（c）分别表示在 5 个节点、8 个节点和 15 个节点这三种情况下，不同网 20 络拓扑结构中算法运算速度的对比。结果表明，对于同一算法，网状结构的网络所消耗的计算时间要远大于其他几种网络拓扑结构所消耗的时间。对于前四种结构来说，其算法运行的时间是网状结构的 60%左右；对于混合结构的网络其算法运行时间是网状结构的 80%左右。

进一步分析可以发现，在不同的算法中，随着网络节点间连接数的增加，算法运行所消耗的时间也随之增加，但增加的速度趋势呈现出变大的趋势，如图 所示。这是由于四种算法运算的基础是分支界限树，无论何种网络结构，在节点相同的情况下，其分支界限树的结构是相同的，有区别的只是节点间的连接数，即节点间约束集的大小不同。当节点数和节点间连接数较小时，算法中的大部分时间用于构建分支限界树，在节点数相同的情况下，分支限界树构建速度相同，而此时节点间连接数较少且相差不多，因此算法在各种结构中运算速度区别不是特别明显。而随着节点数增加，网状结构的节点间连接数急速增加，此时算法的时间更多用于分支界限树的遍历和方案约束值的计算，因此算法运行时间会大幅增加。

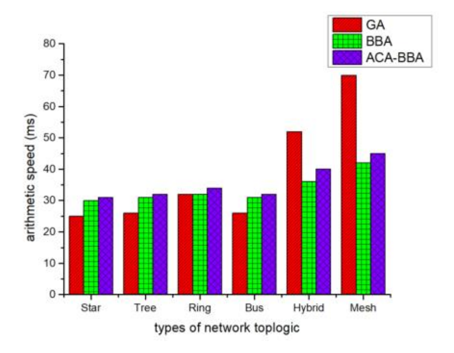
4.3 不同算法运行效率测试与分析

为了验证各种算法的运行速度，对比算法之间优化效率，本小节进行了详细的测试。本文将从三个方面进行测试：a)测试算法的运算速度及其变化；b)测试算法优化效率和变化规律；c)测试优化效率与算法运行速度、剪枝率之间的关系。以建立的模拟网络结构和环境为基础，对本文中提到的四个算法进行了实验和测试。通过对模拟网络进行计算，对各个算法的性能进行了测试，主要的测试指标为

a）算法运行时间。针对不同网络环境和网络结构，算法运行时所消耗的时间，以 ms 为单位。

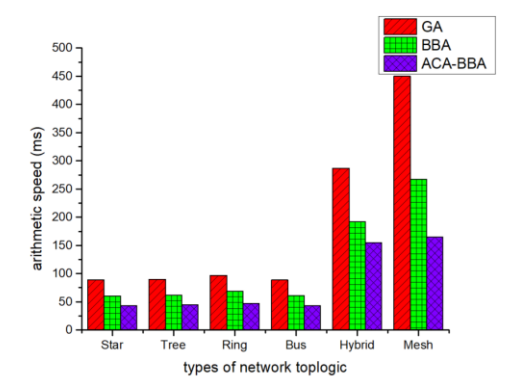
b）剪枝率。计算被剪掉的分支占所有分支的比例算法优化效率。

c）优化程度。分别对比 BBA 和 GA、ACA-BBA 和 BBA 的运行速度，计算算法优化的比例。



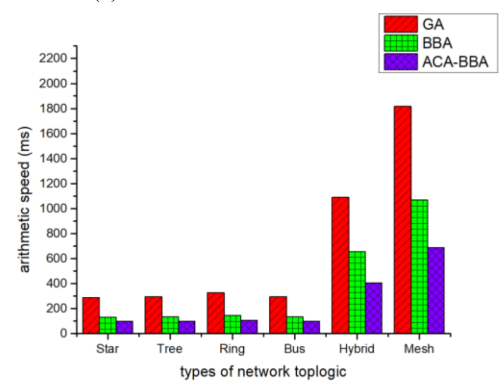
(a) 网络中包含 5 个节点

(a) Network includes 5 nodes: trace 1



(b) 网络中包含 8 个节点

(b) Network includes 8 nodes: trace 2



(c) 网络中包含 15 个节点

(c) Network includes 15 nodes: trace 3

图 5 在不同网络结构中运算速度的对比图

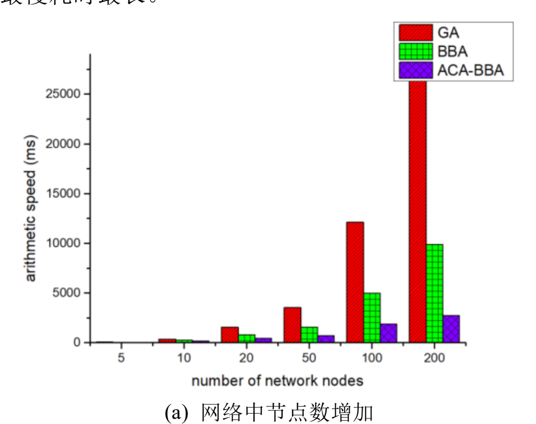
Fig. 5 The arithmetic speed in different topologies.

为了描述网络的复杂性，本文引入了节点连接率的概念，节点连接率等于图中边的数目与全连接时边的数目之比，以百分数的形式体现。以混合型网络为基础，随机增加节点和节点间的连接，并为每对相连节点随机分配约束值，模拟真实网络结构和网络环境。在本实验中，为了保证算法的简单与快速，本文设置网络节点数为 5~100，节点间的连接率为 20%（节点数小于 10 时，节连接率设为星形节点的连接率）。这些参数的设置来自于上一节分析的普通网络结构中有近 80%左右的不同类型节点数和网络连接率都在此范围内。由于篇幅限制，关于参数的自适应调整和分析可作为下一步的工作，不在本文中论述。

4.3.1 算法运算速度测试与分析

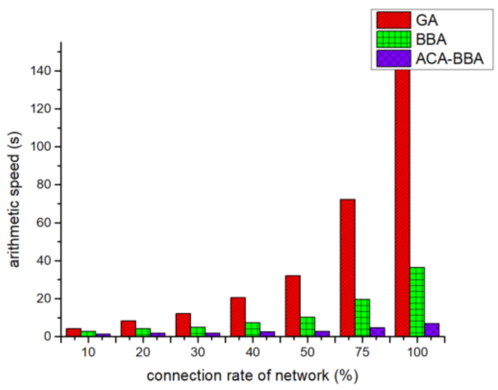
各个算法的运算速度如图 6 所示，图中三个算法对于同一网络结构进行运算所消耗时间的对比。图 6（a）（b）分别表示在网络节点数增加、节点连接率增加这两种情况下，各个算法运行所消耗时间的对比。结果表明，对于同一种网络

结构，运算所消耗时间 ACA-BBA 最低，GA 用时最多运算速度非常慢。对于节点数不增加连接率增加的情况，三个算法所需时间均有不同程度提高，特别是对于 GA 算法其运行时间大幅增加，而其余两个算法运行时间也有所增加，但增加幅度相对不如 GA 那么明显；对于节点数增加连接率不增加的情况，三种算法的运算速度也有所降低，其中 GA 算法速度最慢耗时最长。



(a) 网络中节点数增加

(a) Increasing number of network nodes: Trace 1



(b) 网络节点间的连接数增加

(b) Increasing connection rate of network nodes: Trace 2

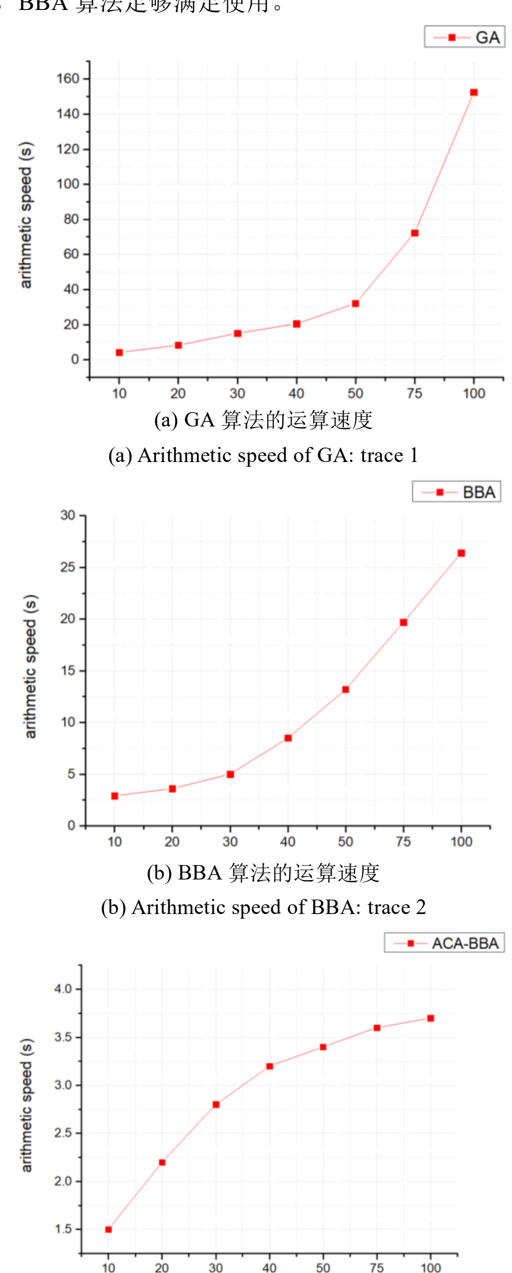
图 6 运算速度的对比图

Fig. 6 The arithmetic speeds of 3 algorithms.

为了对比这三个算法，本文进一步分析了实验的结果，如图 7 所示。在节点数保持 100，连接率增加的情况下，连接率从 10%增加到 50%时，BBA 和 ACA-BBA 算法耗时增加较快，但当连接率超过 50%后，其增加幅度放缓。这是由于连接率的增加，提高了分支限界算法计算的复杂度，但这两个算法都会对分支限界树进行剪枝，当剪枝率达到一定程度后，即使再增加节点间的连接，由于剪枝率的提高，计算速度增涨的幅度会有所降低。相比之下 GA 算法由于不进行剪枝操作，其计算所消耗时间会随连接率升高而不断增加，且增速越来越快。在节点数增加，连接率不变或者增加的情况下三种算法所消耗时间均会增加，但增加幅度不同，这是由于三个算法剪枝率不同，没有剪枝的 GA 算法增速最快，剪枝率高的 ACA-BBA 算法增速最慢。

4.3.2 算法剪枝率测试与分析

在这一节中，分别测试并分析了 BBA 和 ACA-BBA 两个算法在对分支界限树的运算过程中优化的效率。其结果如图 8 所示，展示了在不同的节点数和节点连接率下，BBA 算法和 ACA-BBA 算法过程中对于对分支界限树的剪枝率（即剪去的分支占所有分枝的比例）。从实验结果中可以看出，ACA-BBA 算法在剪枝率仍然有明显的优势：随着节点数或连接率的增加，ACA-BBA 算法的剪枝率在不断提高，而 BBA 算法的剪枝率相对比较稳定，一般在 30%~50%。只有当节点数小于 10 时，ACA-BBA 算法的剪枝率会略低于 BBA 算法。但由于在现实环境中小于 10 个节点的网络其计算所需时间不高，且小于 10 个节点的网络中其安全设备部署问题不会太复杂，BBA 算法足够满足使用。



(a) GA 算法的运算速度

(a) Arithmetic speed of GA: trace 1

(b) BBA 算法的运算速度

(b) Arithmetic speed of BBA: trace 2

（c) ACA-BBA 算法的运算速度

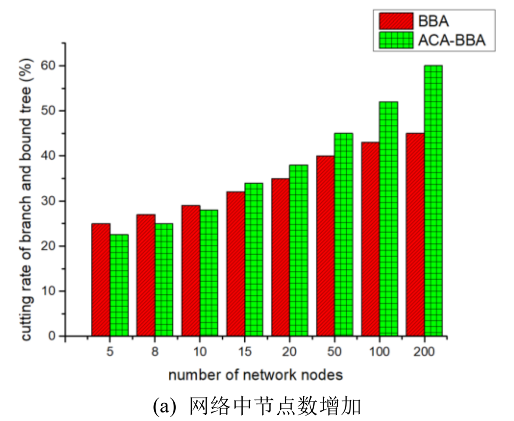
（c) Arithmetic speed of ACA-BBA: trace 3

图 7 不同节点连接率时各个算法的运算速度

Fig. 7 Arithmetic speed of different algorithms in different cutting rates of branch and bound tree.

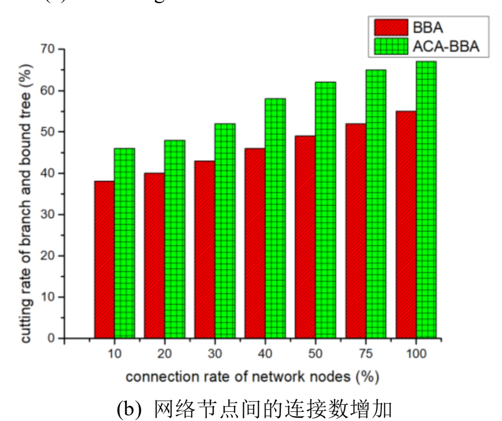
对于节点数不变连接率增加的情况，从直观上来看，连接率的增加确实使得分支界限树的计算变得更为复杂，使得 BBA 和 ACA-BBA 两个算法的剪枝率在不同程度上发生了变化。在 ACA-BBA 算法中剪枝率与弧一致预处理技术的优化效果有很大的关系，不同的连接情况下，预处理的优化效果会有所不同。具体来说，网络中在节点连接具备分布不均衡特征的情况下，弧一致预处理技术的优化效果很可能由于以下两个原因而有所降低：a）网络中某一部分节点间连接率非常高，即网络中部分节点连接非常密集时，连接非常密度节

点的约束值会使弧一致预处理算法计算得到的最小约束发生偏差，最终影响优化效果;b）网络中的连接率较高且分布非常平均，即网络中每个节点的连接数几乎相同时，分支界限树中每条路径的约束值比较接近，使用弧一致预处理算法得到的最小约束值不能发挥很好剪枝效果。这两种情况在实际的网络环境中出现的概率较低，且即使在这两种情况下，ACA-BBA 算法的剪枝率依然高于 BBA 算法。因此综合来看，ACA-BBA 算法的优势是非常明显的。



(a) 网络中节点数增加

(a) Increasing number of network nodes: trace 1



(b) 网络节点间的连接数增加

(b) Increasing connection rate of network nodes: trace 2

图 8　剪枝率的对比图

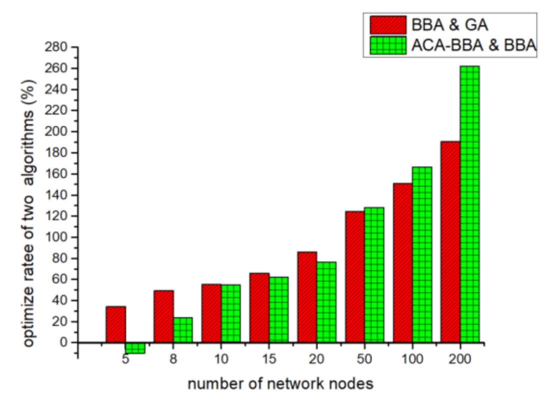
Fig. 8　Cutting rate of branch and bound tree.

4.3.3 算法优化效率测试与分析

本文分别对比 BBA 和 GA、ACA-BBA 和 BBA 的运行速度，对算法优化率进行了统计，结果如图 9 所示。

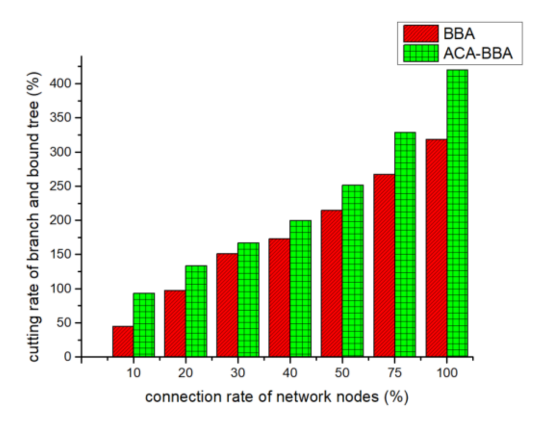
图 9（a）表示在节点连接率不变、节点数增加的情况下，三种算法两两之间的优化率。从图中可以发现，当节点数较少时（小于 10 个节点左右），两种算法优化的比率较小，这是因为节点较少时算法的大部分时间用于构建分支界限树，计算约束开销集所消耗时间较少，因此算法优化效率不高；当节点数继续增加时，BBA 算法和 ACA-BBA 算法的优势逐渐体现出来，其优化率随之逐渐升高，且增长速度越来越快。

图 9（b）表示在节点数不变、节点连接率增加的情况。从实验结果可以看到，BBA 算法和 ACA-BBA 算法有着较为明显的优化效果：当连接率小于 20%时，BBA 和 ACA-BBA 的优化效果均不高于 30%；当连接率高于 20%后，优化率明显提高，且增涨速度非常快。但由于本文在实验中使用了 100%的连接率来进行计算，即每两个网络节点之间均存在直接连接，因此对于有 n 个节点的网络，其连接数为 $n\times(n-1)/2$，在真实的网络环境是不可能有如此多的直接连接，因此这也只是理论值。优化率之所以快速的增加的原因在于，BBA 和 ACA-BBA 算法进行了剪枝。当连接率提高后，很多增加连接就直接被剪去，运算速度不会随连接率的增加而大幅降低。



(a) 网络中节点数增加

(a) Increasing number of network nodes: Trace 1



(b) 网络节点间的连接数增加

(b) Increasing connection rate of network nodes: Trace 2

图 9　算法优化率对比图

Fig. 9　Optimize ratio of 3 algorithm.

5　结束语

本文研究基于帕累托优化下的网络安全设备部署问题。首先提出针对多目标条件下的分布式约束优化问题，使用七元组来形式化描述网络空间安全问题。随后基于分支界限算法构建部署方案搜索算法在解空间内进行搜索，求出最佳部署方案。为了提高运算速度，本文基于弧一致技术构建数据预处理优化算法，对分支界限搜索算法进行优化，以达到在含有大量节点的解空间内进行快速搜索的目的。最后通过仿真实验验证了部署方案搜索算法和优化算法的正确性和有效性。

参考文献：

[1] 崔承刚，杨晓飞. 基于内部罚函数的进化算法求解约束优化问题[J]. 软件学报, 2015, 26(7): 1688-1699. (Cui Chenggang, Yang Xiaofei. Interior penalty rule based evolutionary algorithm for constrained optimization [J]. Journal of Software, 2015, 26(7): 1688−1699)

[2] 张晓薇，曹东刚，陈向群，等. 一种网络化移动应用部署方案优化方法 [J]. 软件学报，2011，22(12):2866-2878. (Zhang XiaoWei, Cao DongGang, Chen XiangQun,　et al. Deployment solution optimization for mobile network applications [J]. Journal of Software, 2011, 22(12): 2866-2878)

[3] 袁佳歆，陈柏超，贾嘉斌. 基于遗传算法的逆变器控制规律 [J]. 电力系统自动化, 2004, 18(24) , 32-35. (Yuan Jiaxin, Chen Baichao, Jia Jiabin. Genetic algorithm based approach for inverter control [J].

Automation of Electric Power Systems, 2004, 18(24) 32-35.)

[4] 袁佳歆，赵震，费雯丽,等. 基于免疫算法的逆变器多目标 Pareto 最优最优控制策略 [J]. 电子技术学报，2014，29(12): 33-41. (Yuan Jiaxin, Zhao Zhen, Fei WENli *et al.* Pareto optimum control of invreter based on multi-objrctive immune algorithm [J]. Transactions of China Electrotechnical Society. 2014, 29(12): 33-41)

[5] 胡旺，张鑫. 基于 Pareto 熵的多目标粒子群优化算法 [J]. 软件学报 2014, 25(5): 1025-1050. (Hu Wang, Zhang Xin. Multi-objective particle swarm optimization based on pareto entropy [J]. Journal of Software, 2014, 25(5): 1025-1050)

[6] 张建峰. 网络安全态势评估若干关键技术研究 [D]. 长沙:国防科学技术大学,2013. (Zhang Jianfeng. Research on several key technologies of cyber security situation assessment [D]. Changsha: National University of Defense Technology, 2013.)

[7] Fave F M D, Stranders R, Rogers A, et al. Bounded decentralised coordination over multiple objectives [C]//Proc of the 10th International Conference on. Autonomous Agents and Multiagent Systems.2011: 371–378.

[8] Fuchs C. Implications of deep packet inspection (DPI) Internet surveillance for society [C]//The Privacy Security Research Paper Series, Privacy & Security Research Paper.2012.

[9] Hayes C, Kesan J. At war over cispa: Towards a reasonable balance between privacy and security. Illinois program in law [C]//Behavior and Social Science Paper.2012.

[10] Herzog S. Revisiting the estonian cyberattacks: digital threats and multinational responses [J]. Journal of Strategic Security, 2011,4(2): 49–60,.

[11] Marinescu R. Exploiting problem decomposition in multi-objective constraint optimization [C]//Proc of the 15th International Conference on Principles and Practice of Constraint Programming.2009:592–607.

[12] Matsui T, Silaghi M, Hirayama K, et al. Distributed search method with bounded cost vectors on multiple objective dcops [C]//Proc of the 15th International Conference on Principles and Practice of MultiAgent Systems. 2012:137–152.

[13] Bringmann K, Friedrich T, Neumann F, et al. Approximation-guided evolutionary multi-objective optimization [C]//Proc of the 22nd International Joint Conference on Artificial Intelligence, 2011: 1198–1203.

[14] Wang Lingyu, Zhang Mengyuan, Jajodia S, et al. Modeling network diversity for evaluating the robustness of networks against zero-day attacks [C]//Proc of European Symposium on Research in Computer Security. 2014:494-511.

[15] Condron S M. Getting it right: Protecting american critical infrastructure in cyberspace [J]. Journal of Law Technology, 2007,20(2): 403–422,.

[16] De Vries K. Proportionality overrides unlimited surveillance the german constitutional court judgment on data retention[C]// CEPS Papers in Liberty and Security in Europe.2010.

[17] Deb K, Agrawal S, Pratap A, et al. A fast and elitist multiobjective genetic algorithm: NSGA-II [J]. IEEE Trans on Evolutionary Computation, 2002, 6(2): 182–197.